Data integration in a Hadoop-based data lake: A bioinformatics case

Article *in* International Journal of Data Mining & Knowledge Management Process · July 2022

DOI: 10.5121/ijdkp.2022.12401

3 authors:

Julia Couto
Pontifícia Universidade Católica do Rio Grande do Sul
18 PUBLICATIONS **71** CITATIONS
SEE PROFILE

Olimar Teixeira Borges
Pontifícia Universidade Católica do Rio Grande do Sul
12 PUBLICATIONS **57** CITATIONS
SEE PROFILE

Duncan D. Ruiz
Pontifícia Universidade Católica do Rio Grande do Sul
111 PUBLICATIONS **914** CITATIONS
SEE PROFILE

International Journal of Data Mining & Knowledge Management Process (IJDKP), Vol.12, No.4, July 2022

Data integration in a Hadoop-based data lake: A bioinformatics case

Júlia Colleoni Couto, Olimar Teixeira Borges, and Duncan Dubugras Ruiz

School of Technology, PUCRS University,
julia.couto, olimar.borges[@edu.pucrs.br], duncan.ruiz@pucrs.br

Abstract. When we work in a data lake, data integration is not easy, mainly because the data is usually stored in raw format. Manually performing data integration is a time-consuming task that requires the supervision of a specialist, which can make mistakes or not be able to see the optimal point for data integration among two or more datasets. This paper presents a model to perform heterogeneous in-memory data integration in a Hadoop-based data lake based on a top-k set similarity approach. Our main contribution is the process of ingesting, storing, processing, integrating, and visualizing the data integration points. The algorithm for data integration is based on the Overlap coefficient since it presented better results when compared with the set similarity metrics Jaccard, Sørensen-Dice, and the Tversky index. We tested our model applying it on eight bioinformatics-domain datasets. Our model presents better results when compared to an analysis of a specialist, and we expect our model can be reused for other domains of datasets.

Keywords: Data integration, Data lake, Apache Hadoop, Bioinformatics.

1 Introduction

Data integration is a challenging task, even more nowadays where we deal with the V's for big data, such as variety, variability, and volume (Searls [1]; Lin et al.[2]; Alserafi et al. [3]). Regarding the variety of data to be integrated into data lakes, having different types of data can be considered one of the most difficult challenges, even more because most datasets may contain unstructured or semi-structured information (Dabbèchi et al. [4]). According to Hai, Quix, and Zhou [5], it is very onerous to perform interesting integrative queries over distinct types of datasets. Another challenge is the high-dimensionality data that may be stored in the data lake. To compute the similarity for that high-dimensional data is expensive. Checking whether the tables are joinable or not is time-consuming because of the large number of tables that may have in a data lake (Dong et al. [6])

To manually analyze different datasets for data integration, a person must check the attributes and at least a dataset sample. To perform a more elaborated work, the person must look for the data dictionary of each dataset, and sometimes it is not easily available. According to Sawadogo and Darmont [7], it is a problem since it is time-consuming, error-prone, and can lead to data inconsistency. Among the methods for data integration, the logic-based ones that consider the dataframes as sets, such as the based on the overlap of the values, could provide useful solutions (Levy [8]).

In previous work [9], we developed a model to perform heterogeneous data integration, taking advantage of a data lake we build based on Hadoop. In this paper we extend this previous work by deepening the explanation of the the methodology we followed, on how we help in solving the challenges related to big data profiling, and by comparing the current work with the state of art. We also deepen the explaining regarding the data ingestion process.

The integration model is based data profiling and schema matching techniques, such as row content-based overlapping. To do so, having defined the datasets for the experiments,

DOI:10.5121/ijdkp.2022.12401

related to the domain of bioinformatics, we build a data lake to ingest, store, process, and visualize the data. We use Apache Nifi for data ingestion and the HDFS (Hadoop Distributed File System) for data storage. We process the data using Python, and we create visualizations of the data using Neo4J.

Our main contribution is a model that allows to quickly ingest different kinds of textual datasets, transform them into dataframes, and, using an approach based on in-memory set similarity for data integration, we suggest the top-k points of integration for the data. We present experiments with eight bioinformatics datasets, and we compare our approach with manual data integration performed by a domain specialist. Our paper can also be used as a guide to building a data lake from scratch.

2 Background

In this section, we briefly present the basic concepts related to our study. We summarize data lake, data profiling, data integration, and present the datasets we used in our experiments. We finish by discussing the related work and present our conclusions.

2.1 Data lake

In a previous work (Couto et al. [10]), we define a *data lake* as a central repository for raw data storage, processing, and analysis, that can be used for unstructured, semi-structured, and structured datasets. A data lake can be composed of different software with its own tasks in an integrated ecosystem. It means we can have different software for data ingestion, storage, processing, presentation, and security, and they have to work together. The most used tool to create a data lake is Apache Hadoop [10]. Forster [11] states that Hadoop is the most used distributed platform for storage, processing, and analysis of big data. Hadoop is an Open-Source Software (OSS) developed in Java and maintained by the Apache Software Foundation [12]. Hadoop is based on the Google MapReduce paradigm and in Google File System, and it is mainly used for distributed processing in computer clusters. We populate our data lake with bioinformatics datasets.

2.2 Data Profiling

According to [13], data profiling is the process of metadata discovery. [14] also states there are three main types of data profiling tasks, namely single-column tasks, multi-column tasks, and dependency detection. Single-column tasks are related to domain classification, cardinalities, values distributions, patterns and data types. Multi-column tasks present correlations and association rules, clusters and outliers, summaries and sketches. In the dependency detection category, there are three main tasks, namely uniqueness (key discovery, conditional, and approximate), inclusion dependencies (foreign key discovery, conditional, and approximate), and functional dependencies (conditional and approximate). The *approximate* items are methods that present approximate results for the proposed tasks.

2.3 Data integration

Data integration deals with the problem of combining different data sources to provide the user with a unified view (Lenzerini [15]). There are different approaches for data integration, and we base our work on the top-k overlap set similarity problem: For all the attributes in all the dataframes, find the top fits for data integration, according to the

intersection among the attributes' distinct values (Zhu et al. [16]). We based our work on an in-memory set similarity approach and the integration is executed using Python notebooks. As similarity metrics, we use the most well-known distance measures for sets similarity according to Ontanón (2020) [17]: Tverski's (Tversky [18]), Sørensen's index (Sørensen [19]), and Jaccard (Jaccard [20]), compared to the Szymkiewicz-Simpson overlap coefficient (Vijaymeena and Kavitha [21]).

2.4 Bioinformatics datasets

Bioinformatics is the product of the union of computer science and biology (Lesk [22]), where we use software to make inferences about datasets of modern molecular biology, so we can connect the data and extract valuable predictions. There are a lot of bioinformatics datasets available, having the most variate information, formats, types, and size. Our study selected eight datasets to populate our data lake and work on automatized data integration. Table 1 presents the characteristics of each dataset, ordered by size from the smaller (DRUGBANK) to the larger (IID). Table 1 also shows that we selected heterogeneous datasets, having varied sizes (from 1 MB to 1,8GB), from 13k entries to almost 1 million entries, with the number of attributes varying from 6 to 253. The datasets are also available in different formats, such as TXT, XML, TSV, JSON, and via API.

Table 1. Characteristics of the bioinformatics datasets [9]

Dataset	Size (MB)	Entries	Attributes	Format
DRUGBANK	0,95	13580	9	XML
DRUGBANK PROTEIN	1,40	26965	7	XML
OMIM	1,80	17092	14	TXT
DRUGCENTRAL	2,50	17390	19	TSV
MONDO	4,00	43233	12	JSON
DISGENET	10,30	84037	16	TSV
UNIPROT	30,20	565255	7	API
REACTOME	37,90	826877	6	TXT
IID	1800,00	975877	253	TXT

- OMIM (McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD), 2021 [23]): Online Mendelian Inheritance in Man - human genes and genetic phenotypes. We used the *genemap2* dataset.
- DISGENET (Pinero et al. [24]): Collections of genes and variants associated with human diseases.
- REACTOME (Jassal et al. [25]): We are using the *UniProt2Reactome* dataset. It is composed of reactions, proteins, and pathways.
- MONDO (Mungall et al. [26]): Ontology for disease definitions.
- DRUGBANK (Wishart et al. [27]): Pharmaceutical knowledge base, we split it into two dataframes: DRUGBANK and DRUGBANK_PROTEIN.
- IID (Kotlyar et al. [28]): Integrated Interactions Database - database of detected and predicted protein-protein interactions. We used the human data annotated dataset.
- DRUGCENTRAL (Avram et al. [29]): Online drug compendium - we use the drug-target interaction dataset.
- UNIPROT (Consortium [30]): We are using the *reviewed Swiss-Prot XML* dataset, a non-redundant and manually annotated database containing protein sequences.

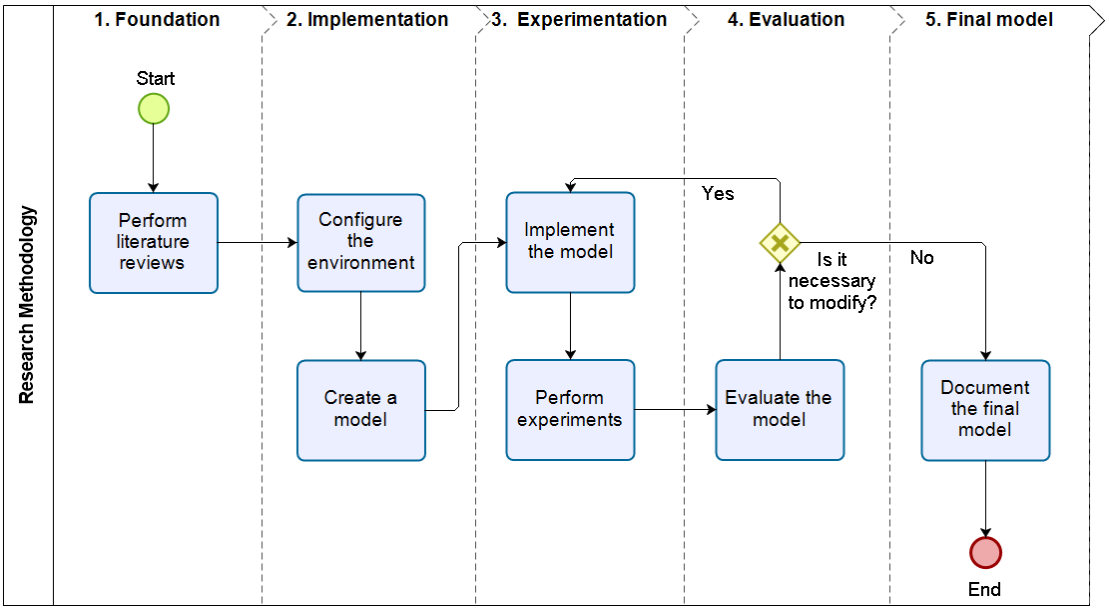

Fig. 1. Research flow

3 Research Design

This section details each step of the method we created to help us achieve our objective. We classified our research as experimental regarding technical procedures. In experimental research, the researchers insert new experimental variables into the environment and perform measures [31]. In our case, we add a data integration model in a Hadoop-based data lake and measure the results, comparing it to a specialist evaluation. We designed a method composed of five stages: Foundation, Implementation, Experimentation, Evaluation, and Final Model. We present the overall flow in Figure 1.

3.1 Foundation

In the Foundation Phase, we *perform literature reviews*: one mapping study (MS) [10] and two systematic literature reviews (SLR) [32, 33], which we previously published. We started by performing an MS to explore and understand how the data lake concept had evolved through time and also what are the techniques and tools that compose the most used architectures related to data lakes. Then, we perform an SLR to delimit our search and understand how people perform big data profiling because profiling was little explored in the previous MS. Finally, we performed an SLR to deepen the knowledge about data integration, and to map the most related studies. In this paper, we present how we face the challenges we identified in the previous literature reviews.

3.2 Implementation

During the Implementation Phase, we configure the computational environment, choose the tools and techniques we use, and create a model for data integration in Hadoop-based data lakes. To *Configure the environment*, we select and prepare a software and hardware infrastructure to create a data lake, using bioinformatics datasets in a Linux-based environment. Next, we *create a model* responsible for data ingestion, storage, processing, and visualization.

3.3 Experimentation

In the Experimentation Phase, we design the algorithm for data integration, and we implement and test the model by the use of Python programming language. The *implement the model* task comprehends the actual development and tests of the system to manage data integration in Hadoop-based data lakes, besides data ingestion, storage, processing, and visualization. The next task is *to perform experiments* to test how our model can be compared with the baseline.

3.4 Evaluation

In the Evaluation Phase, we present how we asses our model. We *evaluate the model* against the manual mapping performed by a user specialist, and we present the runtime according to a scalability test. We also present the challenges we approach compared to the state-of-the-art presented in the literature.

3.5 Final Model

In the last Phase, we *document the final model* alongside all the processes that allows us to create the model.

4 Related work

In previous work, we performed two Systematic Literature Review (SLR) about big data profiling [33] and data integration in data lakes [32]. In the SLR about data profiling, we generate an overview of data profiling in big data, focusing on the challenges related to the field. Then, we perform an SLR about data integration in data lakes, where we identify the trending topics, and the most recent works closely related to our model, and the open challenges.

In this section, we compare our work with the related papers that the found performing the SLRs, regarding the challenges in big data profiling and in data integration in data lakes. We performed an analysis to check which challenges are already addressed by which papers (see Table 2 and Table 3). We read the papers searching for the challenges keywords to perform this investigation. Then we performed an overall reading to check if the papers really do not address the challenges.

In the SLR about big data profiling, we disregard the papers about literature reviews [34–36] to compare only the papers that present solutions (frameworks, algorithms, or software) for big data profiling. We also remove the challenge *lack of research* from the Table, since all published papers help to address this challenge. If we recap the results of the SLR about data profiling results, we identify that an optimal solution would address all the 15 challenges. As we explain next, we can see our model creating alternatives to help mitigate them.

1. Complexity: we contribute to this challenge by working with data preparation on diverse types of data.
2. Continuous profiling: using Apache Nifi scheduling process, we can ingest the data-sets according to a predefined amount of time and then use HDFS and Python to reprocess the dataframes and rerun the model.
3. Incremental profiling: See *Continuous profiling.*
4. Interpretation: our model allows us to understand and interpret data profiling results by using it for data integration.

Table 2. Papers versus challenges they address - RSL Big Data Profiling. Adapted from [33]

Papers	**Challenges**	Complexity	Continuous profiling	Incremental profiling	Interpretation	Metadata	Online profiling	Poor data quality	Profiling dynamic data	Topical profiling	Value	Variability	Variety	Visualization	Volume
Our work	**13**	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Canbek et al. (2018) [37]	9				✓	✓		✓		✓	✓	✓	✓	✓	✓
Ardagna et al. (2018)[38]	7							✓	✓	✓	✓	✓	✓		✓
Chrimes and Zamani (2017) [39]	7	✓			✓	✓				✓	✓			✓	✓
Koehler et al. (2019) [40]	6	✓				✓		✓			✓		✓		✓
Sampaio et al. (2019) [41]	6	✓						✓		✓			✓	✓	✓
Vieira et al. (2020) [42]	6	✓						✓		✓			✓	✓	✓
Santos et al. (2019) [43]	5	✓									✓		✓	✓	✓
Alserafi et al. (2016) [3]	4					✓		✓				✓	✓		
Liu et al. (2013) [44]	4	✓					✓		✓	✓					
Maccioni and Torlone (2017) [45]	4			✓		✓	✓							✓	
Taleb et al. (2019) [46]	4					✓					✓			✓	✓
Khalid and Zimányi (2019) [47]	3	✓				✓								✓	
Jang et al. (2018) [48]	2							✓			✓				
Sun et al. (2018) [49]	2									✓				✓	
Shaabani and Meinel (2018) [50]	1					✓									
Heise et al. (2013) [51]	1													✓	

5. Lack of research: we contribute by creating a model to help mitigate existing challenges.
6. Metadata: we do not create metadata, but we use existing metadata as input for our model.
7. Online profiling: our model does not directly address this challenge.
8. Poor data quality: we contribute to the field of bioinformatics data quality by preprocessing the datasets to be processed by our model.
9. Profiling dynamic data: See *Continuous profiling*.
10. Topical profiling: we work on the specific topic of bioinformatics datasets.
11. Value: the data integration model we developed can help users make better decisions based on the suggested data integration points.
12. Variability: we work with data that vary regarding size, content, type, and other aspects.
13. Variety: we profile heterogeneous textual data to work on data integration.
14. Visualization: our model provides visualization aid to help understand the data integration created by data profiling techniques.
15. Volume: although the datasets we used in our experiments are not quite huge in volume, we believe our model can escalate for larger datasets if we provide more computational resources than we had available.

Regarding the SLR about data integration in data lakes, we also mapped some related challenges. In the same way in the previous analysis, we omit the challenge *lack of available solutions* from the Table, since all published papers also help in addressing this challenge. Table 3 presents the relationships among the challenges and the papers we selected in the second SLR. Next, we discuss how we believe our model contributes to help mitigating the challenges.

Table 3. Papers versus challenges they address - RSL Data integration in Data Lakes. Adapted from [32]

Papers	**Challenges**	Complexity	Computational cost	Diversity	In-memory integration	Non-generalizable solutions	Scalability	Variability	Variety
Our work	8	✓	✓	✓	✓	✓	✓	✓	✓
Jovanovic et al. (2021) [52]	6	✓	✓		✓	✓		✓	✓
Kathiravelu and Sharma (2017) [53]	5	✓		✓	✓		✓		✓
Dong et al. (2021) [6]	5	✓	✓		✓	✓	✓		
Endris et al. (2019) [54]	5	✓	✓		✓	✓			✓
Alrehamy and Walker (2018) [55]	5	✓				✓	✓	✓	✓
Zhu et al. (2019) [16]	5	✓	✓		✓	✓	✓		
Hai et al. (2018) [5]	4	✓		✓		✓	✓		
Alserafi et al. (2016) [3]	4			✓		✓		✓	✓
Yang et al. (2020) [56]	3		✓			✓	✓		
Quinn et al. (2020) [57]	3	✓	✓				✓		
Dhayne et al. (2021) [58]	3	✓		✓					✓
Pomp et al. (2018) [59]	2	✓							✓
Brackenbury et al. (2018) [60]	2	✓					✓		
Zhang and Ives (2019) [61]	2			✓		✓			
Dabbèchi et al. (2020) [4]	2			✓					✓
Rezig et al. (2021) [62]	2	✓				✓			
Helal et al. (2021) [63]	1			✓					
Beyan et al. (2016) [64]	1								✓
Koutras (2019) [65]	1					✓			
Alili et al. (2017) [66]	1		✓						
Haller and Lenz (2020) [67]	1								✓
Hai and Quix (2019) [68]	1	✓							

- Complexity: our model ease the complexity of data transformation by ingesting, storing, processing, and visualizing data in an integrated manner.
- Computational cost: we present the experiments that show the computational cost for our model.
- Diversity: our model implements the use of different elements in a data lake, which are easily integrated among them.
- In-memory integration: our model performs in-memory data integration.
- Lack of available solutions: we contribute by creating a model to help mitigate existing challenges.
- Non-generalizable solutions: although we present experiments with bioinformatics datasets, the model can deal with different data domains.
- Scalability: we present the experiments that show the scalability of our model.
- Variability: the user can rerun our model always that the datasets to be analyzed present any changes.
- Variety: our model ingests data from different textual sources and formats.

It is important to note that regarding the papers that are not related to a specific challenge, the authors do not explicitly write about the item - although they also may contribute to the challenge resolution somehow.

By analyzing Tables 2 and 3, we can see that the challenges complexity, variability, and variety are shared in both areas: big data profiling and big data integration. Although variability and variety are part of the V's from big data, we can see that there are still opportunities to improve in those areas. On the other hand, there are challenges specific to big data profiling, such as *continuous profiling*, *incremental profiling*, *profiling dynamic data*, and *topical profiling*. Likewise, some challenges are more specific to data integration, such as *in-memory integration*.

5 Problem Statement

As stated by Khalid and Zimányi [47], managing and querying a data lake is a difficult task, mainly because the data is heterogeneous, may have replicas or versions, have a considerable volume, and present quality issues. In this sense, data integration, which represents 80-90% of the challenges for data scientists (Abadi et al. [69]), is a fundamental task to enable querying a data lake. However, integrating heterogeneous data into data lakes is a complex task, mainly due to the variety of data types (Hendler [70], Alrehamy and Walker [55] that can compose the data lake. If we only talk about textual data, there are countless extensions and possible formatting, such as: .txt, .docx, .csv, .xls, .xml, .json and so on. Furthermore, the analysis for the integration depends on experts, often data scientists, who need to spend time inspecting data profile information, such as the types of each attribute, a sample of that data, or studying the data dictionary - when the dictionary is available. Finally, data integration is essential for extracting a more holistic view and information from the data lake, enabling us to make simple to complex queries and add value to the information.

Therefore, for the problem of automatic data integration in data lakes, the input would a number of heterogeneous datasets and a threshold that limits the integration points of interest. The output would be the points of integration among the datasets, and the evaluation measures would be the ones based on an expert evaluation of the integration points.

Regarding the complexity of the problem, according to Alserafi et al. [3], the equation to calculate the total number of comparisons that needed to be performed to find the columns candidate to data integration is

$$comparisons = \left[d \times \frac{d-1}{2} \right] \times m^2 \tag{1}$$

where d represents the number of datasets, and m represents the average number of attributes for each dataset. Considering our datasets (previously presented in Table 1), we have 344 attributes in total, considering 9 dataframes, then $m = 38$. Thus, we would have to perform about 51984 comparisons among the attributes.

6 Results

The purpose of this study is to create a model for automating the integration of datasets. To do so, we use similarity measures to check the possibility of data integration in a Hadoop-based data lake. We started by creating the *system architecture* for the data lake, based on Docker containers. Then, we worked on *data management*. Finally, we present the *algorithm* we developed.

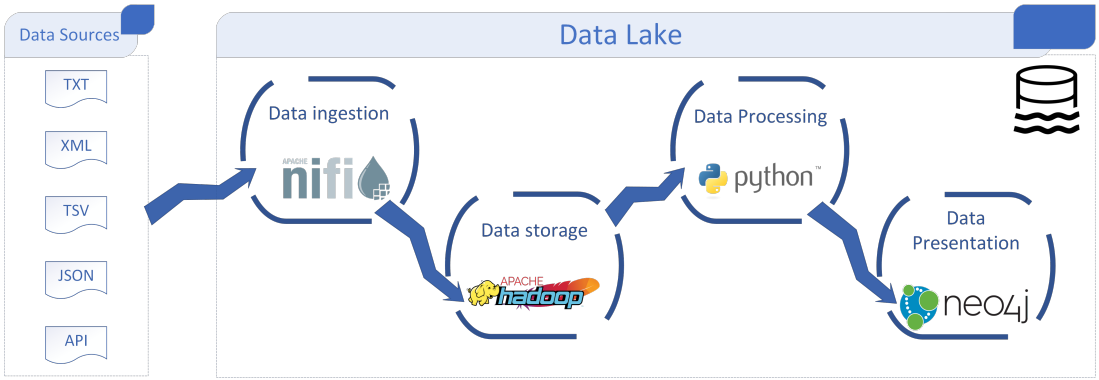

Fig. 2. Composition of the data lake [9]

6.1 System architecture

Our data lake is supported by an Ubuntu 20 64-bit Linux server, having the following configuration: 16GB RAM DDR3, Processor Intel® Core(R) I7-4790 CPU@3.6GHz x 8, 1TB disk capacity. The data lake is composed of ten Docker containers:

1. Apache Nifi: a framework used for data ingestion.
2. Python - Jupyter Notebook: a programming language and a web application to run Python code, used for data processing.
3. Neo4J: a graph database used to visualize the integration among the dataframes.
4. Hadoop Namenode: the master node in the HDFS architecture
5. Hadoop History Server: keeps the logs of all the jobs that ran on Hadoop.
6. Hadoop Resource Manager: contains the YARN (Yet Another Resource Negotiator), a service that manages the resources and schedules/monitors the jobs.
7. Hadoop Node Manager: launches and manages the containers on a node.
8. Hadoop Datanodes: Three containers (Datanode1, Datanode2, Datanode3). The worker's nodes in the HDFS architecture, where the data is stored.

6.2 Data management

Data management includes data ingestion, storage, processing, and presentation, as illustrated in Figure 2. We ingested data into the data lake by creating processes in Apache Nifi. We create one process for each dataset, where the process searches for the dataset on HTTP (for MONDO, REACTOME, DISGENET, IID, and DRUGCENTRAL), or in a local folder (for OMIM and DRUGBANK, because they are not available directly due to the need of registering and licensing). Then we unzipped some of the datasets, and we renamed them all for standardization. Apache Nifi then sends the datasets to Hadoop, where they are stored in HDFS. For UNIPROT, we use the API directly on Jupyter. Then, we start data processing using the Python - Jupyter Notebook docker. Lastly, we create a graph visualization, based on Neo4J, to present the results.

6.3 Data Ingestion & Storage

To be able to integrate the dataframes, we first need to ingest and store the data inside the data lake. Since data ingestion and storage are closely related, we present both processes together. We ingested data into the data lake by creating processes in Apache Nifi. We

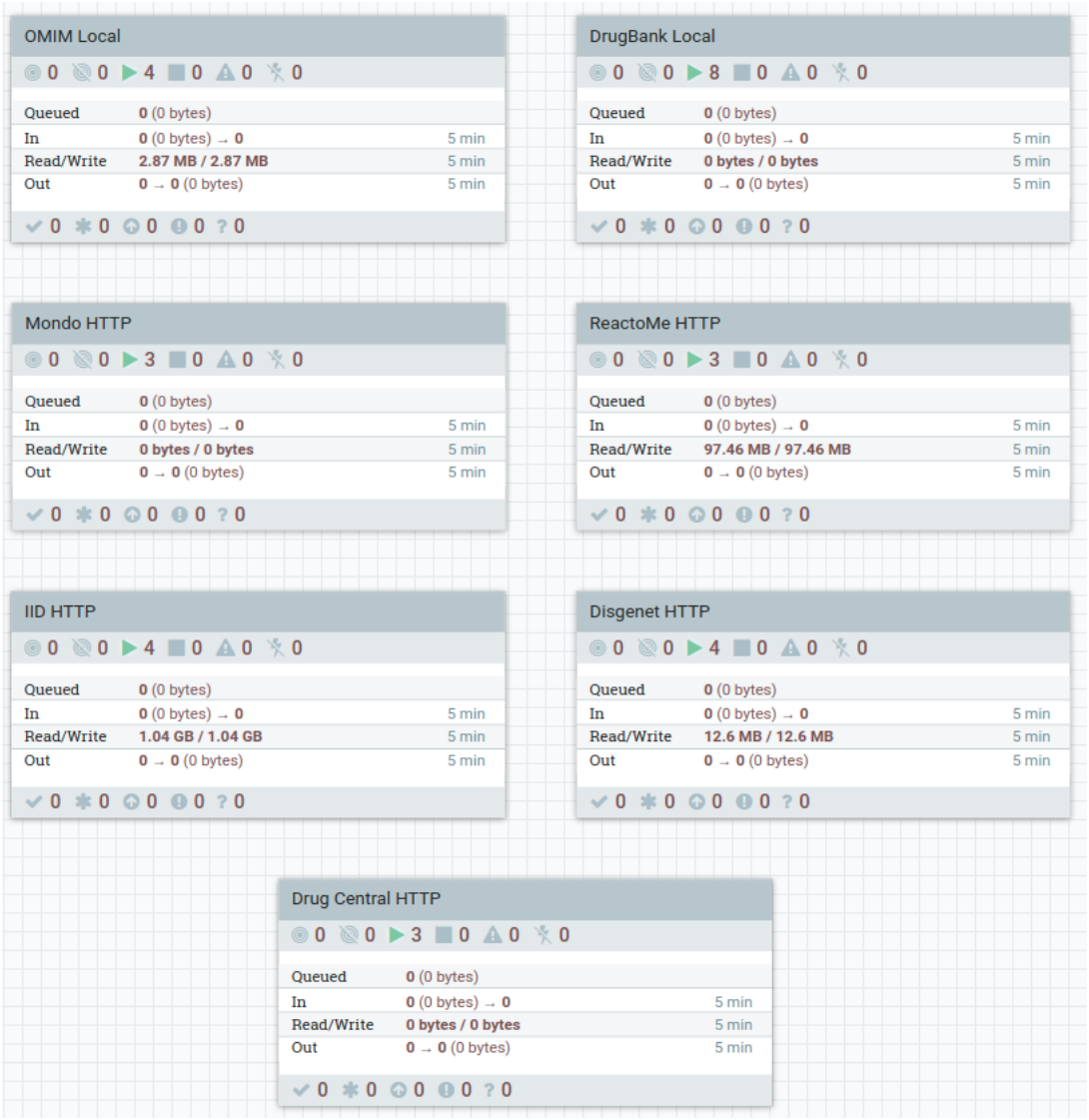

Fig. 3. Groups of processors created on Apache Nifi for data ingestion

create one group of processes for each dataset, where the process searches for the dataset on HTTP (for MONDO, REACTOME, DISGENET, IID, and DRUGCENTRAL), or in a local folder for OMIM and DRUGBANK because they are not available directly due to the need of registering and licensing. Then we unzipped some of the datasets and renamed them all for standardization. Next, Apache Nifi sends the datasets to Hadoop to be stored in HDFS. For UNIPROT, we use the API directly on Jupyter notebooks.

Figure 3 presents the home page when accessing Apache Nifi, having the seven processes groups responsible for getting our datasets into Hadoop. We collect the datasets using different Nifi processors, detailed above:

- ListFile: Fetches the files from the local filesystem. For this processor, we configure the properties:
  - – Input Directory: /datasets
  - – Input Directory Location: Local
  - – File Filter: the name of the dataset (OMIM_genemap2.txt)

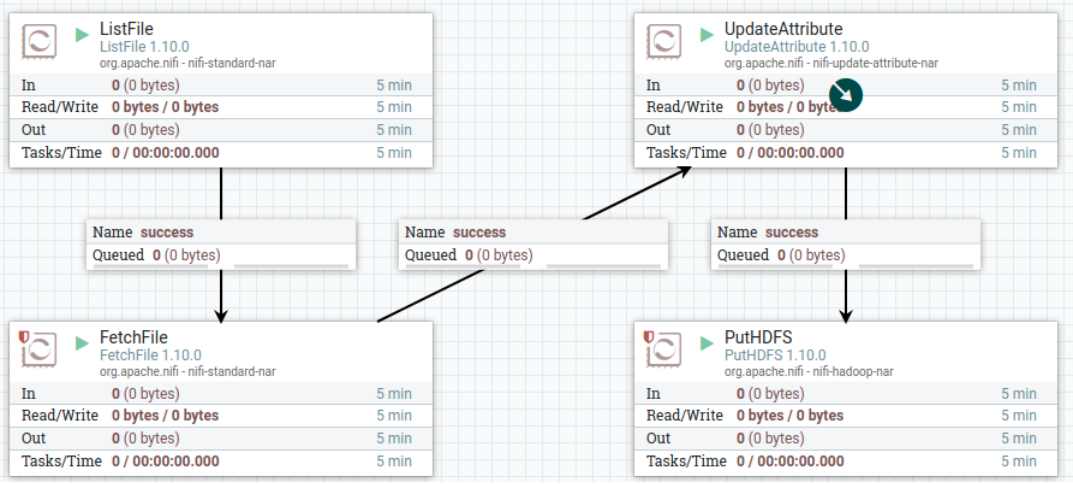

Fig. 4. Apache Nifi - OMIM group of processors

- InvokeHTTP: Connects with a HTTP Endpoint. For this processor, we configure the properties:
  - – HTTP Method: GET
  - – Remote URL: URL to the file
- FetchFile: Reads the contents of a file and streams it.
- UnpackContent: Unpacks the FlowFile contents.
  - – Packaging Format: zip (also accepts tar and flowfiles)
- CompressContent: Decompresses or compresses FlowFile contents based on user-specified algorithms. The properties we configure are:
  - – Mode: decompress
  - – Compression Format: gzip (also accepts bzip2, xz-lzma2, lzma, snappy, etc.)
- UpdateAttribute: Updates or deletes attributes based on a regular expression. We configure the property:
  - – filename: name of the file to be stored in Hadoop.
- PutHDFS: Writes the FlowFile data to HDFS. In this processor we set the following attributes:
  - – Hadoop Configuration Resources: /etc/conf/core-site.xml
  - – Directory: /datasets
  - – Conflict Resolution Strategy: replace. Other options include *ignore*, *fail*, or *append*.

Figure 4 presents an example of a process group for the OMIM dataframe. We can also specify the Nifi Schedule to get the file for all the processors. For instance, we can use scheduling parameters to get the files once per day, once per week, or each n hours. Below we present the list of processors we used for each dataset.

- OMIM: ListFile → FetchFile → UpdateAttribute → PutHDFS
- DRUGBANK: ListFile → FetchFile → UnpackContent → UpdateAttribute → PutHDFS
- MONDO, REACTOME, DRUGCENTRAL, and DRUGBANK (json): InvokeHTTP → Update-Attribute → PutHDFS
- IID and DISGENET: InvokeHTTP → CompressContent → UpdateAttribute → PutHDFS

We store the data in Hadoop HDFS. We configure Hadoop to have three datanodes for data replication. It means that, in a Hadoop traditional architecture, each data block is replicated on each of the datanodes, which helps speed up the data processing.

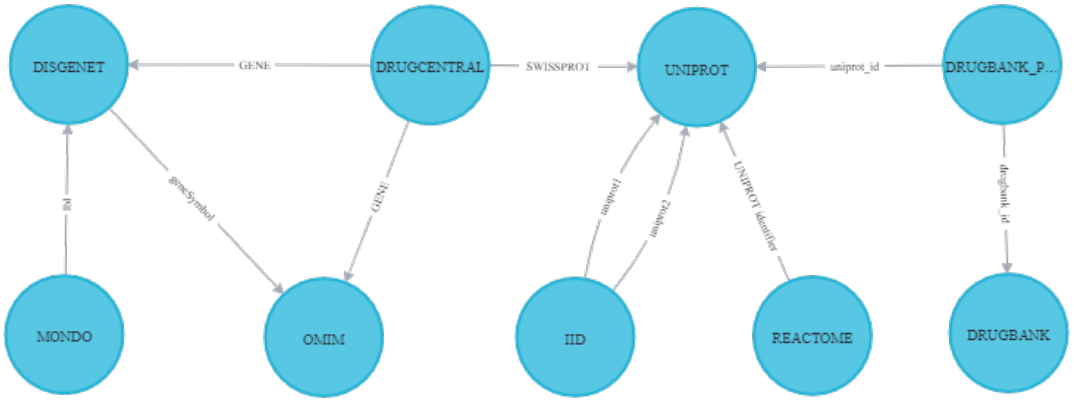

Fig. 5. Manually mapped integration [9]

6.4 Data Processing & Integration

We start the experiments by turning the datasets into Python Pandas dataframes. Pandas is a Python library for data analysis and manipulation. The standardization of the datasets as dataframes assure a unified entry for the algorithm, solving issues regarding one dataset being derived under one condition and the others being on other conditions. To create the DRUGBANK dataframe, we based on the solution provided by [71]. We also use other libraries, such as HDFS, that provide a pure HDFS client, bioservices that provide API access to UNIPROT, and the package *py_stringmatching* that implements the similarity metrics.

After creating the dataframes, one of the authors, a specialist in data science, analyzed the datasets to manually map the attributes candidates for points of integration. To do so, the specialist analyzed the names of the columns and a sample of data for each column, using data profiling techniques. The specialist took about four hours to finish this analysis, and we present the manual mapping in Figure 5. Figure 5 presents the manual data integration points, based on a graph visualization, where the nodes or vertices are the names of the dataframes, and the edges are the attributes' names. The orientation of the arrow indicates that, for instance, the attribute 'lbl' from the Mondo dataframe is a point of integration to the dataframe Disgenet, meaning that a percentage of 'lbl' is also present in another attribute of Disgenet. We developed this Figure to be later compared with the results of the algorithm we developed for data integration so that we could compare a user specialist analysis with the algorithm output.

Our algorithm is based on the concept of intersection or overlap between two attributes in sets of data. We first identify the unique values of each attribute for each dataset. Then we compare each dataset attribute with all the other datasets' attributes to check if the unique values of the content of each attribute are contained in any other attributes of all of the other datasets. The attribute with fewer unique values indicates the orientation of the data integration. For instance, let us analyze the following case that includes dataframes (df) and attributes (att):

- df1['att01'] has 10 unique values;
- df2['att06'] has 20 unique values;
- 10 values from df1['att01'] are also present on df2['att06'].

In that case, we can notice that 100% of df1['att01'] are also present in df2['att06'], being that a good point for data integration. The notation would be: df1['att01'] $\longrightarrow$

df2['att06']. Regarding the minimum value for data intersection, we defined a threshold of 0.4 (in a range from 0–1)] to identify good integration points, but it is configurable according to the user's needs. It means that if two columns in a dataframe have 40% or more of data in common, the two columns are candidates for data integration, and the dataframes where the columns come from are integrable.

To define the best threshold for our experiments, we tested different values and compared the results with the specialist's analysis. We started with 0.9, and after each execution, we compared our results with the specialist's manual integration. When we noticed that the selected threshold retrieved at least all of the integration points defined by the specialist, we stopped decreasing the threshold, determining the value of 0.4.

Figure 6 details the activities diagram for the Algotrithm 1 we developed. Figure 6 shows that we can configure restrictions to select the attributes to be analyzed, such as the minimum number of unique values that an attribute must have to enter in the comparisons, and if we want to perform comparisons with attributes that contain only numeric values. Other restrictions include: removing attributes with only nulls or NaN and removing attributes with binary values (0 or 1, T or F). The binary values would not present real candidate points for data integration among the datasets since they mostly represent True or False values. For instance, the IID dataset presents dozens of attributes that are named after diseases, where the value = 0 corresponds to False and values = 1 corresponds to True (e.g.: 'bone disease', 'overnutrition', 'asthma', 'lymphoma').

Algorithm 1 Pseudo-code for the data integration algorithm

Require: $_1$*datasets*, $_2$`minimum number of unique values`, $_3$`accept numeric comparisons` (*True or False*)
Ensure: `dataframeLeftName + columnName, dataframeRightName + columnName: Overlap:`*value* / `Jaccard:`*value* / `Sørensen-Dice:`*value* / `Tversky:`*value*

```
for dataframes do
    while columns in the dataframe = True do
        if compare numeric values = False then
            if value is numeric = True then
                next column in while
            end if
        end if
        if unique values ≤ predetermined value OR target column has already been compared = True then
            next column in while
        else
            if column values ≠ binary values then
                for dataframes + 1 do    ▷ repeats the same logic as the previous for for the next dataframe
                    ... for
                    if minimum number of unique values between compared columns ≠ 0 then
                        calculate Overlap, Jaccard, Sørensen-Dice, and Tversky
                        if Overlap, Jaccard, Sørensen-Dice, and Tversky > 0.4 then
                            return Output = Ensure
                        end if
                    end if
                end for
            end if
        end if
    end while
end for
```

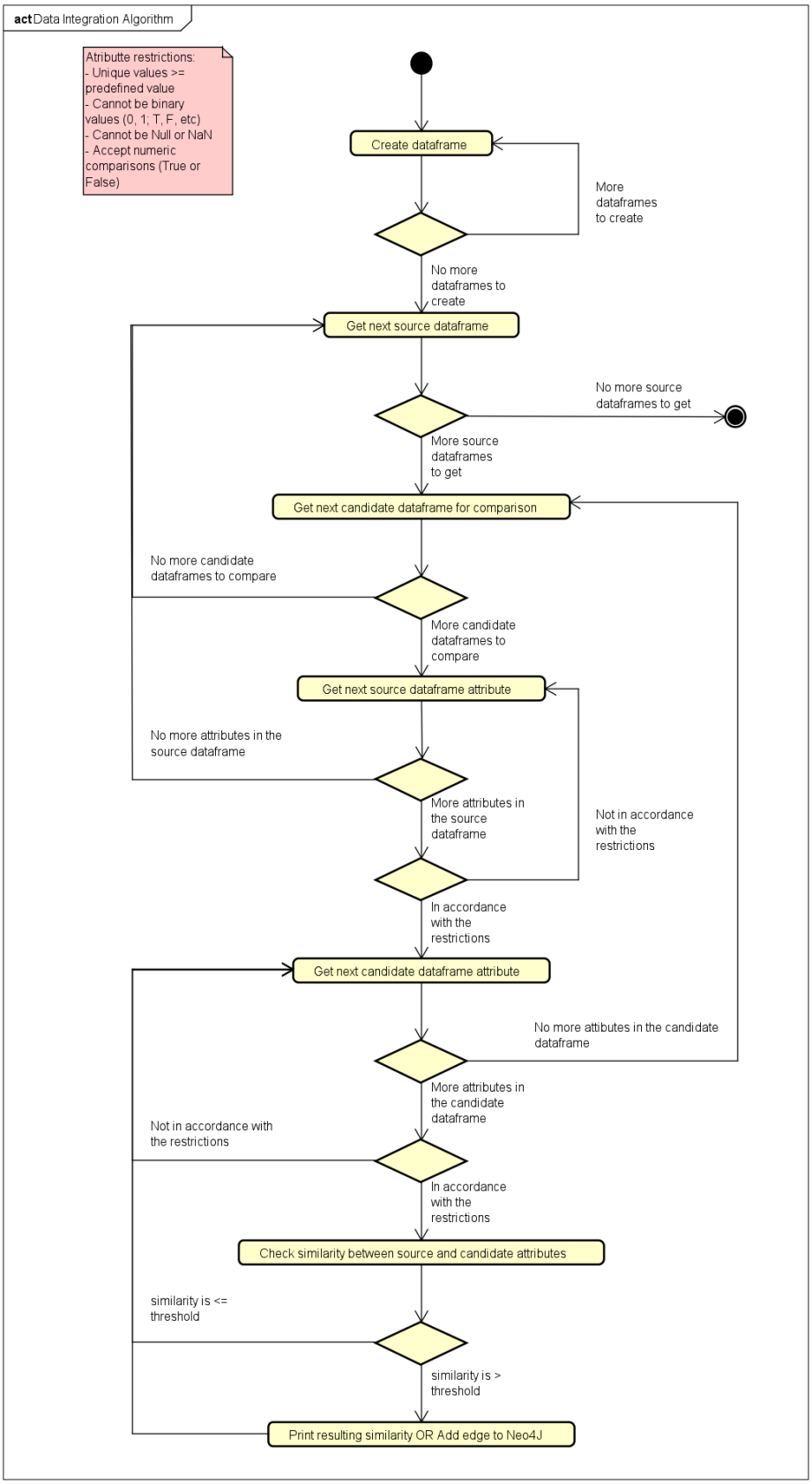

Fig. 6. Algorithm for data integration in UML Activity Notation (Adapted from [9])

The algorithm starts by creating dataframes for all the datasets, then it selects the first source dataframe, which will be compared to the other dataframes. Then the attributes of the source dataframe are compared with the attributes of the first candidate dataframe to check the similarity. It happens until we do not have more source dataframes to be compared to the candidates.

Our algorithm also handles so that there are no redundant comparisons among dataframes and attributes. Firstly, we assure that a dataframe is not compared to itself by identifying its previously defined name in the algorithm. Secondly, when we compared each attribute of the first dataframe, we stored its description in a variable. Before comparing the dataframe's attribute with another, we check that there are no attributes with the same description. Therefore, we exclude the possibility of redundant comparisons between dataframes and attributes.

The algorithm returns a list having the names of the dataframes, attributes, and resulting values for the Szymkiewicz-Simpson overlap coefficient – Equation 2, which is the main result, compared to other similarity metrics (Jaccard – Equation 3, Sørensen-Dice – Equation 4, and Tversky – Equation 5). The resulting values for the similarity metrics range from 0 (attributes are not at all similar) to 1 (attributes contain the same data). Next, we present the equations related to the metrics, where X represents the attribute of the source dataframe and Y represents the attribute of the dataframe candidate to be compared.

The Overlap Equation calculates the size of the intersection divided by the smaller of the size of the two attributes or sets:

$$overlap(X, Y) = \frac{|X \bigcap Y|}{min(|X|, |Y|)} \tag{2}$$

The Jaccard measures the size of the intersection between two sets divided by the size of the union:

$$jaccard(X, Y) = \frac{|X \bigcap Y|}{|X \cup Y|} \tag{3}$$

The Sørensen-Dice similarity score returns twice the intersection divided by the sum of the cardinalities.

$$dice(X, Y) = \frac{2 \times |X \bigcap Y|}{|X| + |Y|} \tag{4}$$

The Tversky index is a generalization of the Sørensen-Dice's and the Tanimoto coefficient (aka Jaccard index) coefficient, but introduces the use of the parameters α and β, where $\alpha = \beta = 1$ produces the Tanimoto coefficient and $\alpha = \beta = 0.5$ produces the Sørensen–Dice coefficient:

$$tversky(X, Y) = \frac{|X \bigcap Y|}{|X \bigcap Y| + \alpha|X - Y| + \beta|Y - X|}; \alpha, \beta >= 0 \tag{5}$$

Our model also presents the option to insert nodes and edges in a Neo4J database to better visualize the relationships among the dataframes.

7 Results

After analyzing the first results presented by the algorithm, we identified that some suggested integration points are numeric values that, in our dataframes, do not represent actual data integration points. For instance:

- UNIPROT['Lenght'] it is the length of the canonical sequence and it varies from 3 to 4 numeric chars;
- OMIM['Entrez_Gene_ID'] the National Center for Biotechnology Information (NCBI) gene ID, values from 1 to 115029024;
- DRUGCENTRAL['STRUCT_ID'] the structure ID, and has values from 1 to 5390;
- DISGENET['YearInitial'] and DISGENET['YearFinal'] are years from 1924 to 2020;
- DISGENET['NofPmids'] the PubMed id, and has values from 1 to 67;
- DISGENET['NofSnps'] the Single nucleotide polymorphisms (SNP) id, has values from 1 to 284.

Because of that, we decided to add a parameter in the algorithm do set if we want to make numeric comparisons. We set the parameter to false since, in our case, it does not represent actual data integration points. However, to be able to generalize for different domains and different types of datasets, that kind of comparison may be useful.

Regarding the similarity metrics, as Tversky, Sørensen-Dice, and Jaccard present correlated values for our data (Tverski's index with α and $\beta = 0.5$ was equal Sørensen-Dice and twice the Jaccard coefficient), we show only the Jaccard and the overlap values in Table 4.

After carefully comparing the Jaccard and Overlap results with the manual mapping and reviewing the actual dataframes, we identified that the Overlap provides better insights about the relationships that could be created among the dataframes.

For instance, for the relationship DISGENET["diseaseType"] and MONDO ["lbl"], the Jaccard index is equal to zero, while the Overlap is 0,667. We checked the data, and we really found a point for integration in that case. Another example is DRUGBANK["drugbank_id"] and DRUGBANK_PROTEIN["drugbank_id"], which represent the same data according to the Overlap coefficient and to our manual analysis, and in this case, the Jaccard index is 0,579.

Hence, the Overlap coefficient seems to represent better how similar two attributes are and the level of data integration we could achieve if we integrate two dataframes using the top-ranked pairs of attributes indicated by the algorithm. Thus, we decided that in our case, it is better to use the Overlap Coefficient.

To better visualize the relationships between the dataframes, we create a database on Neo4J, where the nodes are the dataframes, and the edges are the name of the attributes. Figure 7 presents the final data integration resulting from our algorithm for the bioinformatics dataframes. In this graph visualization, similar to the manually mapped data integration visualization, the names of the dataframes are the vertices. The names of each attribute responsible for the integration are presented in the edges that connect the vertices. Figure 7 presents 9 nodes and 32 edges, some with only one edge between them (such as MONDO and DISGENET) and others having a high concentration of edges, such as IID, UNIPROT, and DRUGCENTRAL. The higher concentration of edges pointing to a dataframe means that the dataframe is referenced by a high number of other dataframes, meaning they represent an important point of integration.

When we manually mapped the points for integration, we identified 10 points (see Figure 5), while our model identifies 32 points, presented in Table 4. For instance, we manually mapped an integration between DRUGCENTRAL["SWISSPROT"] and UNIPROT ["ENTRY"], but our model shows that the coefficient for that integration is less than 0,4, while suggesting two better points: DRUGCENTRAL["GENE"] $\rightarrow$ UNIPROT[" Gene names "], with a overlap of 0,487, and DRUGCENTRAL["ACCESSION"] $\rightarrow$ UNIPROT["ENTRY"], with an overlap of 0,829.

Table 4. Final data integration mapping [9]

DF1	Column DF1	DF2	Column DF2	Overlap	Jaccard
UNIPROT	Entry	← REACTOME	UNIPROT identifier	0,409	0,058
UNIPROT	Gene names	← OMIM	Approved_Symbol	0,466	0,015
UNIPROT	Gene names	← DISGENET	geneSymbol	0,483	0,009
UNIPROT	Gene names	← IID	symbol2	0,484	0,017
UNIPROT	Gene names	← DRUGCENTRAL	GENE	0,486	0,002
UNIPROT	Gene names	← IID	symbol1	0,488	0,017
DRUGCENTRAL	ACCESSION	→ DRUGBANK_PROTEIN	uniprot_id	0,488	0,018
UNIPROT	Organism	← DRUGBANK_PROTEIN	organism	0,499	0,206
DRUGCENTRAL	ACCESSION	→ IID	uniprot1	0,509	0,072
DRUGCENTRAL	ACCESSION	→ IID	uniprot2	0,510	0,071
DISGENET	geneSymbol	← DRUGCENTRAL	GENE	0,528	0,110
REACTOME	UNIPROT identifier	← DRUGBANK_PROTEIN	uniprot_id	0,546	0,030
REACTOME	UNIPROT identifier	← IID	uniprot2	0,565	0,107
REACTOME	UNIPROT identifier	← IID	uniprot1	0,567	0,105
DRUGBANK_PROTEIN	uniprot_id	→ IID	uniprot1	0,583	0,147
DRUGBANK_PROTEIN	uniprot_id	→ IID	uniprot2	0,590	0,147
OMIM	Approved_Symbol	← DRUGCENTRAL	GENE	0,609	0,082
DRUGCENTRAL	GENE	→ IID	symbol1	0,615	0,076
DRUGCENTRAL	GENE	→ IID	symbol2	0,617	0,075
REACTOME	UNIPROT identifier	← DRUGCENTRAL	ACCESSION	0,624	0,019
DISGENET	diseaseType	→ MONDO	lbl	0,667	0,000
REACTOME	Species	→ DRUGCENTRAL	ORGANISM	0,750	0,051
UNIPROT	Entry	← DRUGCENTRAL	ACCESSION	0,829	0,004
OMIM	Approved_Symbol	→ IID	symbol1	0,866	0,704
DISGENET	geneSymbol	→ IID	symbol1	0,880	0,464
OMIM	Approved_Symbol	→ IID	symbol2	0,885	0,717
DISGENET	geneSymbol	→ IID	symbol2	0,898	0,469
UNIPROT	Entry	← DRUGBANK_PROTEIN	uniprot_id	0,909	0,008
DISGENET	geneSymbol	→ OMIM	Approved_Symbol	0,915	0,536
UNIPROT	Entry	← IID	uniprot1	0,953	0,030
UNIPROT	Entry	← IID	uniprot2	0,960	0,031
DRUGBANK	drugbank_id	← DRUGBANK_PROTEIN	drugbank_id	1,000	0,579

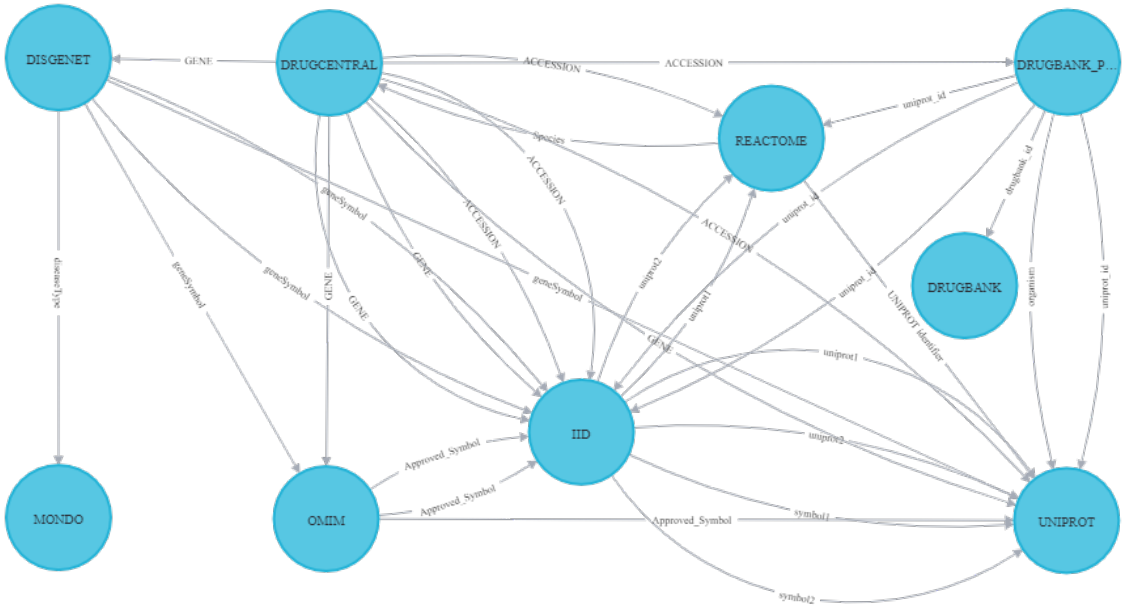

Fig. 7. Final data integration [9]

Additionally, our model discovered 22 more paths of integration that were not manually identified, which we list above:

- From IID and: DISGENET, DRUGCENTRAL, OMIM, REACTOME, and DRUGBANK_PROTEIN
- From UNIPROT and: OMIM and DISGENET
- From REACTOME and: DRUGBANK_PROTEIN and DRUGCENTRAL
- From DRUGBANK_PROTEIN and DRUGCENTRAL

8 Evaluation

Regarding the evaluation, our previous literature review identified that the related work usually evaluates their approaches based on scalability, execution time, precision, recall, F1, and accuracy. To be able to evaluate our experiments based on precision, recall, F1, and accuracy, we would have to have a gold standard, which indicates the points for integration for the datasets we selected. As we did not find another work that analyzed the same datasets for data integration, we manually mapped the integration points as explained before. Therefore, if we calculate those metrics based on our results and the manual mapping, we would have the maximum resulting value for each metric since our model identified all of the manually mapped integration points and even others that were not identified on the manual mapping.

Therefore, we performed an analysis to answer the following question: *1) What is the average execution time and scalability provided by our model, according to the number of dataframes to be analyzed?*. We ran the model 10 times to get the average running time, starting with the bigger dataframe (IID) and ending with the smaller (DRUGBANK) - see Table 1 to check the datasets ordered by size. Figure 8 shows the results of the scalability evaluation. We started the scalability evaluation with only two dataframes; we added one more dataframe each time and checked the runtime. It takes on average 117 minutes to

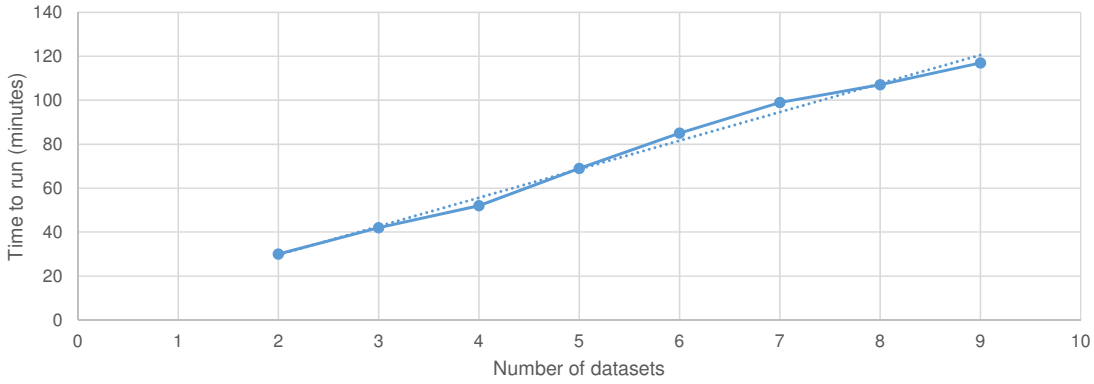

Fig. 8. Scalability - Execution time

run for all dataframes in the hardware we described in Subsection 6.1. Note that we run it in memory, in hardware with a humble configuration.

The scalability of the proposed solution takes place in terms of enabling comparisons between all attributes of all datasets. For example, the 19 attributes of DRUGCENTRAL are compared with the 253 attributes of the IID and so on, creating a bigger and bigger search space as we add more datasets for comparison.

9 Discussion

We faced some challenges during the development and execution of our model. Initially, we had to elaborate on different ways of treating the datasets, as they had different data types. After this process, the researchers met to define the best way to carry out the comparison process. Effectively, the algorithm creation process started when we defined the four ways to calculate distances (Overlap, Jaccard, Sorensen, and Tversky's). Then, the initial algorithm implemented worked for most columns of the datasets. However, the algorithm generated errors, specifically for columns with information of the "JSON" or "XML" type, being corrected and treated soon afterward. After running the algorithm, we noticed that some of the comparisons generated 100% matches in many cases. Therefore, we verified that there were columns with information of binary values, which meant "False" or "True", but that was not necessarily relevant and similar to each other. We address this issue by removing columns with these data types from our comparison. We also skip Null and empty values, in the comparison steps. Furthermore, when we ran with all the datasets simultaneously, the initial version of the algorithm worked but took longer than we expected. Therefore, we performed refactoring in the algorithm, so we executed in the settings described in Section 4.1, we could obtain better results in a considerably shorter time.

The challenges we faced during algorithm development are all data-related. When we start data analysis with data pre-processing, the data must go through a cleaning phase, which could have ruled out some of the related challenges. However, one of the goals of the algorithm is to receive data from different formats with different types of attributes and be able to perform the necessary initial comparisons. In this way, we allow the algorithm to be executed even by people without specific knowledge in data processing, so they can and still obtain good results for their data integration.

Let us now discuss the utility of our model, by considering the data integration example in the field of bioinformatics. A data scientist has access to a data lake with the

same bioinformatics datasets we worked on: OMIM, DISGENET, REACTOME, MONDO, DRUGBANK, IID, DRUGCENTRAL, and UNIPROT. The data scientist received the task to study neglected diseases, such as tuberculosis. To do so, it is necessary to explore the data related to the gene *inhA*, which is related to the organism *Mycobacterium tuberculosis*. Having those two pieces of information, it is easy to find the related data on UNIPROT. Actually, it may be on the top-5 results of a quick search on Google. But how will the person know if and how the data found in UNIPROT can be integrated with the other data sources, so they can find additional information? Well, usually the person would have to put an effort into understanding the schema of all the datasets, analyze the data dictionary, a sample of data, and so on.

Using our data integration model, we will be able to see that UNIPROT is easily integrated with OMIM, DISGENET, IID, and DRUGCENTRAL by the *gene name*. By integrating with OMIM, we would have more details about genetic phenotypes related to the gene *inhA*; while DISGENET would bring the variants of the genes related to the tuberculosis disease. IID adds information about how a protein related to *inhA* (*Enoyl-[acyl-carrier-protein] reductase [NADH]*) interacts with other proteins. UNIPROT can also be integrated with REACTOME since REACTOME contains a field named *UNIPROT identifier*. Thus, we would have additional information about how the molecules interact in a cell to change the cell or create a certain product; for instance, turn genes on or off.

Additionally, integrating with DRUGCENTRAL would add information about interactions related to the drugs and tuberculosis. The integration with DRUGCENTRAL will allow integration with DRUGBANK, which brings supplementary information about the substance of the drugs and related products. For instance, we will find that *Pretomanid* is a medication for the treatment of tuberculosis. Finally, having the disease type from DISGENET, we could connect with the MONDO ontology, and learn about the different types of the disease, such as endocrine, esophageal, ocular, spinal tuberculosis, and others.

10 Conclusions

In this paper, we presented a model for automatized data integration in a Hadoop data lake, and we present experiments with eight well-known datasets from the bioinformatics domain, having different sizes and formats. We tested the similarity among the dataframes with different similarity measures, and we identified that The Overlap coefficient and Jaccard would be enough for us to validate our proposal.

Because the Overlap coefficient presented better results than the actual data and a specialists analysis, our experiments suggest that the Overlap coefficient is the best option for the in-memory set similarity approach we developed. Based on the Overlap coefficient, we found the top-k overlap set similarity that can help define data integration points for datasets in a data lake. For future work, we plan to implement text similarity strategies to magnify the reach of our results and increase the points for data integration based on semantic and syntactic.

Availability of data and materials

The data that support the findings of this study are available from:

- UNIPROT (UniProtKB - Reviewed (Swiss-Prot)). API available at [72].
- OMIM (genemap2). Available at [23], upon register and request. Release: File generated on 02/07/2020.
- DISGENET: Available at [73]. Release: version 7.0, January 2020.

– DRUGCENTRAL: Available at [74]. Release: 18/09/2020.
– IID: Available at [75]. Release: 2018-11.
– MONDO: Available at [76]. Release: v2021-01-15.
– REACTOME: Available at [77]. Release: Version 75, 07/12/2020.
– DRUGBANK: Available at [78], upon registration. Release: 5.1.8, 2021-01-03.

Restrictions apply to the availability of these data, which were used under license for the current study, and so are not all publicly available. Data are, however, available from the authors upon reasonable request and with permission of the owners, when necessary.

Acknowledgments

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior – Brasil (CAPES)* – Finance Code 001. We also thank Professor Anil Wipat and his team at Newcastle University for the support and advice in the early stages of the project and for providing us with the hardware for first creating the data lake.

References

1. D. B. Searls, "Data integration: challenges for drug discovery," *Nature Reviews Drug Discovery*, vol. 4, pp. 45–58, Jan, 2005.
2. X. Lin, X. Li, and X. Lin, "A review on applications of computational methods in drug screening and design," *Molecules*, vol. 25, pp. 1–17, Mar, 2020.
3. A. Alserafi, A. Abelló, O. Romero, and T. Calders, "Towards information profiling: Data lake content metadata management," in *International Conference on Data Mining Workshops*, (Barcelona, ES), pp. 178–185, IEEE, 2016.
4. H. Dabbèchi, N. Z. Haddar, H. Elghazel, and K. Haddar, "Social media data integration: From data lake to nosql data warehouse," in *International Conference on Intelligent Systems Design and Applications*, (Online), pp. 701–710, Springer, 2020.
5. R. Hai, C. Quix, and C. Zhou, "Query rewriting for heterogeneous data lakes," in *European Conference on Advances in Databases and Information Systems*, (Budapest, HU), pp. 35–49, Springer, 2018.
6. Y. Dong, K. Takeoka, C. Xiao, and M. Oyamada, "Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach," in *International Conference on Data Engineering*, (Chania, GR), pp. 456–467, IEEE, 2021.
7. P. Sawadogo and J. Darmont, "On data lake architectures and metadata management," *Journal of Intelligent Information Systems*, vol. 56, no. 1, pp. 97–120, 2021.
8. A. Y. Levy, *Logic-Based Techniques in Data Integration*, ch. 24, pp. 575–595. Boston, MA: Springer, 2000.
9. J. Couto, O. Borges, and D. Ruiz, "Automatized bioinformatics data integration in a hadoop-based data lake," in *International Conference on Data Mining and Applications*, (Vancouver, CA), p. 16, AIRCC Pulishing Corporation, 2022.
10. J. Couto, O. T. Borges, D. Ruiz, S. Marczak, and R. Prikladnicki, "A mapping study about data lakes: An improved definition and possible architectures," in *International Conference on Software Engineering and Knowledge Engineering*, (Lisbon, PT), pp. 453–458, KSI Research Inc., 2019.
11. R. R. Forster, *Hive on Spark and MapReduce: A methodology for parameter tuning.* Master Thesis, NOVA Information Management School, Lisbon, PT, 2018.
12. Apache Software Foundation, "The Apache Software Foundation." Retrieved from `https://apache.org`, 2021. June, 2021.
13. Z. Abedjan, L. Golab, and F. Naumann, "Data profiling: A tutorial," in *International Conference on Management of Data*, (Chicago, US), pp. 1747–1751, ACM, 2017.
14. Z. Abedjan, L. Golab, and F. Naumann, "Profiling relational data: a survey," *The Very Large Databases Journal*, vol. 24, pp. 557–581, Jun, 2015.
15. M. Lenzerini, "Data integration: A theoretical perspective," in *Symposium on Principles of Database Systems*, (New York, US), p. 233–246, ACM, 2002.
16. E. Zhu, D. Deng, F. Nargesian, and R. J. Miller, "Josie: overlap set similarity search for finding joinable tables in data lakes," in *International Conference on Management of Data*, (Amsterdam, NL), pp. 847–864, ACM, 2019.

17. S. Ontañón, "An overview of distance and similarity functions for structured data," *Artificial Intelligence Review*, vol. 53, pp. 5309–5351, Feb, 2020.
18. A. Tversky, "Features of similarity," *Psychological Review*, vol. 84, p. 327, Mar, 1977.
19. T. J. Sørensen, *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons*. Copenhagen, DK: I kommission hos Ejnar Munksgaard, 1948.
20. P. Jaccard, "Distribution comparée de la flore alpine dans quelques régions des alpes occidentales et orientales," *Bulletin de la Murithienne*, vol. XXXVII, pp. 81–92, Jan, 1902.
21. M. Vijaymeena and K. Kavitha, "A survey on similarity measures in text mining," *Machine Learning and Applications: An International Journal*, vol. 3, pp. 19–28, Mar, 2016.
22. A. Lesk, *Introduction to bioinformatics*. Oxford: Oxford university press, 2019.
23. McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD), "Online mendelian inheritance in man, omim®." Retrieved from `https://www.omim.org`, 2021. June, 2021.
24. J. Piñero, J. M. Ramírez-Anguita, J. Saüch-Pitarch, F. Ronzano, E. Centeno, F. Sanz, and L. I. Furlong, "The disgenet knowledge platform for disease genomics: 2019 update," *Nucleic Acids Research*, vol. 48, pp. D845–D855, Jan, 2020.
25. B. Jassal, L. Matthews, G. Viteri, C. Gong, P. Lorente, A. Fabregat, ..., and P. D'Eustachio, "The reactome pathway knowledgebase," *Nucleic Acids Research*, vol. 48, pp. D498–D503, Jan, 2020.
26. C. J. Mungall, J. A. McMurry, S. Köhler, J. P. Balhoff, C. Borromeo, M. Brush, ..., and D. Osumi-Sutherland, "The monarch initiative: an integrative data and analytic platform connecting phenotypes to genotypes across species," *Nucleic Acids Research*, vol. 45, pp. D712–D722, Jan, 2017.
27. D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, ..., and J. Woolsey, "Drugbank: a comprehensive resource for in silico drug discovery and exploration," *Nucleic Acids Research*, vol. 34, pp. D668–D672, Jan, 2006.
28. M. Kotlyar, C. Pastrello, Z. Malik, and I. Jurisica, "Iid 2018 update: context-specific physical protein–protein interactions in human, model organisms and domesticated species," *Nucleic Acids Research*, vol. 47, pp. D581–D589, Jan, 2019.
29. S. Avram, C. G. Bologa, J. Holmes, G. Bocci, T. B. Wilson, D.-T. Nguyen, ..., and T. I. Oprea, "Drugcentral 2021 supports drug discovery and repositioning," *Nucleic Acids Research*, vol. 49, pp. D1160–D1169, Jan, 2021.
30. U. Consortium, "Uniprot: a worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, pp. D506–D515, Jan, 2019.
31. R. Wazlawick, *Metodologia de pesquisa para ciência da computação*. Rio de Janeiro, BR: Elsevier, 2014.
32. J. Couto and D. Ruiz, "An overview about data integration in data lakes," in *Iberian Conference on Information Systems and Technologies*, (Madrid, ES), p. 6, IEEE, 2022.
33. J. Couto, J. Damasio, R. Bordini, and D. Ruiz, "New trends in big data profiling," in *Computing Conference*, (London, UK), p. 14, Springer, 2022.
34. Z. Abedjan, "An introduction to data profiling," in *Business Intelligence and Big Data*, (Cham, DE), pp. 1–20, Springer, 2018.
35. W. Dai, I. Wardlaw, Y. Cui, K. Mehdi, Y. Li, and J. Long, "Data profiling technology of data governance regarding big data: Review and rethinking," in *Information Technology: New Generations*, (Cham, DE), pp. 439–450, Springer, 2016.
36. S. Juddoo, "Overview of data quality challenges in the context of big data," in *International Conference on Computing, Communication and Security*, (Pamplemousses, MU), pp. 1–9, IEEE, 2015.
37. G. Canbek, S. Sagiroglu, and T. Taskaya Temizel, "New techniques in profiling big datasets for machine learning with a concise review of Android mobile malware datasets," in *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism*, (Ankara, TR), pp. 117–121, IEEE, 2018.
38. D. Ardagna, C. Cappiello, W. Samá, and M. Vitali, "Context-aware data quality assessment for big data," *Future Generation Computer Systems*, vol. 89, pp. 548 – 562, Dec, 2018.
39. D. Chrimes and H. Zamani, "Using distributed data over HBase in big data analytics platform for clinical services," *Comp. and Mathematical Methods in Medicine*, vol. 2017, pp. 1–16, Dec, 2017.
40. M. Koehler, E. Abel, A. Bogatu, C. Civili, L. Mazilu, N. Konstantinou, A. Fernandes, J. Keane, L. Libkin, and N. W. Paton, "Incorporating data context to cost-effectively automate end-to-end data wrangling," *IEEE Transactions on Big Data*, vol. X, pp. 1–18, Apr, 2019.
41. S. Sampaio, M. Aljubairah, H. A. Permana, and P. Sampaio, "A conceptual approach for supporting traffic data wrangling tasks," *The Computer Journal*, vol. 62, p. 461–480, Mar, 2019.
42. A. A. Vieira, L. M. Dias, M. Y. Santos, G. A. Pereira, and J. A. Oliveira, "On the use of simulation as a big data semantic validator for supply chain management," *Simulation Modelling Practice and Theory*, vol. 98, pp. 1–13, Jan, 2020.

43. M. Santos, C. Costa, J. Galvão, C. Andrade, O. Pastor, and A. Marcén, "Enhancing big data warehousing for efficient, integrated and advanced analytics: visionary paper," *Lecture Notes in Business Information Processing*, vol. 350, pp. 215–226, Jun, 2019.
44. B. Liu, H. Chen, A. Sharma, G. Jiang, and H. Xiong, "Modeling heterogeneous time series dynamics to profile big sensor data in complex physical systems," in *International Conference on Big Data*, (Santa Clara, US), pp. 631–638, IEEE, 2013.
45. A. Maccioni and R. Torlone, "Crossing the finish line faster when paddling the data lake with KAYAK," *Very Large Databases Endowment*, vol. 10, pp. 1853–1856, Aug, 2017.
46. I. Taleb, M. Serhani, and R. Dssouli, "Big data quality: A data quality profiling model," *Lecture Notes in Computer Science*, vol. 11517, pp. 61–77, Jun, 2019.
47. H. Khalid and E. Zimányi, "Using rule and goal based agents to create metadata profiles," *Communications in Computer and Information Science*, vol. 1064, pp. 365–377, Sep, 2019.
48. W.-J. Jang, J.-Y. Kim, B.-T. Lim, and G.-Y. Gim, "A study on data profiling based on the statistical analysis for big data quality diagnosis," *International Journal of Advanced Science and Technology*, vol. 117, pp. 77–88, Mar, 2018.
49. H. Sun, S. Hu, S. McIntosh, and Y. Cao, "Big data trip classification on the New York City taxi and Uber sensor network," *Journal of Internet Technology*, vol. 19, pp. 591–598, Feb, 2018.
50. N. Shaabani and C. Meinel, "Improving the efficiency of inclusion dependency detection," in *International Conference on Information and Knowledge Management*, (Torino, IT), pp. 207–216, ACM, 2018.
51. A. Heise, J. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann, "Scalable discovery of unique column combinations," *Very Large Databases Endowment*, vol. 7, pp. 301–312, Dec, 2013.
52. P. Jovanovic, S. Nadal, O. Romero, A. Abelló, and B. Bilalli, "Quarry: a user-centered big data integration platform," *Information Systems Frontiers*, vol. 23, pp. 9–33, Dec, 2021.
53. P. Kathiravelu and A. Sharma, "A dynamic data warehousing platform for creating and accessing biomedical data lakes," in *Very Large Data Bases Workshop on Data Management and Analytics for Medicine and Healthcare*, (Munich, DE), pp. 101–120, Springer, 2017.
54. K. M. Endris, P. D. Rohde, M.-E. Vidal, and S. Auer, "Ontario: Federated query processing against a semantic data lake," in *International Conference on Database and Expert Systems Applications*, vol. 11706, (Bratislava, SK), pp. 379–395, Springer, 2019.
55. H. Alrehamy and C. Walker, "SemLinker: automating big data integration for casual users," *Journal of Big Data*, vol. 5, pp. 1–14, Mar, 2018.
56. Z. Yang, B. Zheng, G. Li, X. Zhao, X. Zhou, and C. S. Jensen, "Adaptive top-k overlap set similarity joins," in *International Conference on Data Engineering*, (Dallas, US), pp. 1081–1092, IEEE, 2020.
57. C. Quinn, A. Z. Shabestari, T. Misic, S. Gilani, M. Litoiu, and J. McArthur, "Building automation system - bim integration using a linked data structure," *Automation in Construction*, vol. 118, p. 16, Oct, 2020.
58. H. Dhayne, R. Kilany, R. Haque, and Y. Taher, "Emr2vec: Bridging the gap between patient data and clinical trial," *Computers & Industrial Engineering*, vol. 156, p. 107236, Jun, 2021.
59. A. Pomp, A. Paulus, A. Kirmse, V. Kraus, and T. Meisen, "Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures," *Technologies*, vol. 6, p. 29, Sep, 2018.
60. W. Brackenbury, R. Liu, M. Mondal, A. J. Elmore, B. Ur, K. Chard, and M. J. Franklin, "Draining the data swamp: A similarity-based approach," in *Workshop on Human-In-the-Loop Data Analytics*, (Houston, US), pp. 1–7, ACM, 2018.
61. Y. Zhang and Z. G. Ives, "Juneau: Data lake management for jupyter," *Very Large Databases Endowment*, vol. 12, p. 1902–1905, Aug, 2019.
62. E. Rezig, A. Vanterpool, V. Gadepally, B. Price, M. Cafarella, and M. Stonebraker, "Towards data discovery by example," *Lecture Notes in Computer Science*, vol. 12633, pp. 66–71, Sep, 2021.
63. A. Helal, M. Helali, K. Ammar, and E. Mansour, "A demonstration of kglac: A data discovery and enrichment platform for data science," *Very Large Databases Endowment*, vol. 14, p. 2675–2678, Jul, 2021.
64. O. D. Beyan, S. Handschuh, A. Koumpis, G. Fragidis, and S. Decker, "A Framework for Applying Data Integration and Curation Pipelines to Support Integration of Migrants and Refugees in Europe," in *Working Conference on Virtual Enterprises*, (Porto, PT), pp. 588–596, HAL, Oct. 2016.
65. C. Koutras, "Data as a language: A novel approach to data integration," in *International Conference on Very Large Database - PhD Workshop*, (Los Angeles, US), pp. 1–4, Springer, 2019.
66. H. Alili, K. Belhajjame, D. Grigori, R. Drira, and H. Ben Ghezala, "On enriching user-centered data integration schemas in service lakes," *Lecture Notes in Business Information Processing*, vol. 288, pp. 3–15, Jun, 2017.
67. D. Haller and R. Lenz, "Pharos: Query-driven schema inference for the semantic web," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, (Wurzburg, GE), pp. 112–124, Springer, 2020.

68. R. Hai and C. Quix, "Rewriting of plain so tgds into nested tgds," *Very Large Databases Endowment*, vol. 12, p. 1526–1538, Jul, 2019.
69. D. Abadi, A. Ailamaki, D. Andersen, P. Bailis, M. Balazinska, P. Bernstein, ..., and D. Suciu, "The seattle report on database research," *SIGMOD Record*, vol. 48, p. 44–53, Dec, 2019.
70. J. Hendler, "Data integration for heterogenous datasets," *Big Data*, vol. 2, pp. 205–215, Dec, 2014.
71. D. S. Himmelstein, "User-friendly extensions of the DrugBank database v1.0." Retrieved from `https://doi.org/10.5281/zenodo.45579`, 2016. November, 2021.
72. UniProt Consortium, "UniProt KB Reviewed (Swiss-Prot) dataset." Retrieved from `https://www.uniprot.org/downloads`, 2021. June, 2021.
73. Integrative Biomedical Informatics Group, "DisGeNET curated gene-disease associations dataset." Retrieved from `https://www.disgenet.org/static/disgenet_ap1/files/downloads/curated_gene_disease_associations.tsv.gz`, 2021. June, 2021.
74. S. Avram, C. G. Bologa, J. Holmes, G. Bocci, T. B. Wilson, D.-T. Nguyen, R. Curpan, L. Halip, A. Bora, J. J. Yang, *et al.*, "DrugCentral dataset." Retrieved from `https://drugcentral.org/download`, 2021. June, 2021.
75. M. Kotlyar, C. Pastrello, Z. Malik, and I. Jurisica, "The iid database." Retrieved from `http://iid.ophid.utoronto.ca/static/download/human_annotated_PPIs.txt.gz`, 2021. June, 2021.
76. OBO Foundry, "Mondo dataset - json edition." Retrieved from `http://purl.obolibrary.org/obo/mondo/mondo-with-equivalents.json`, 2021. June, 2021.
77. Reactome, "Reactome UniProt to pathways dataset." Retrieved from `https://reactome.org/download/current/UniProt2Reactome_All_Levels.txt`, 2021. June, 2021.
78. OMx Personal Health Analytics, Inc., "DrugBank dataset." Retrieved from `https://go.drugbank.com/releases/latest`, 2021. June, 2021.

Authors

J Couto holds a degree in Information Systems (2012), an MBA in Project Management (2016), a Master's in Computer Science (2018), and a Ph.D. in Computer Science from PUCRS (2022). She works as a professor at PUCRS university, and as a data engineer at NoHarm.ai. Her main research focus is on automating the integration of big data based on data profiling.

O. T. Borges is a Ph.D. student at PUCRS. He holds a degree in Information Systems (2015) and a Master's in Computer Science (2018). He is a member of the MuNDDoS Research Group (Distributed Software Development Research Group). His current research focus is supporting software development using Artificial Intelligence and Machine Learning techniques to Software Startups.

D. D. Ruiz holds a BS in Electrical Engineering from UFRGS (1983), a master's degree (1987), and a Ph.D. (1995) in Computer Science from the same university (UFRGS) and post-doctorate in Computer Science at Georgia Institute of Technology (2002). He is is a professor in the School of Technology at PUCRS, Brazil, where he leads the GPIN research group, acting primarily in the core of Machine Intelligence and Robotics. He has been working mainly in the areas of business process automation, non-conventional databases, bioinformatics, and database knowledge discovery (KDD).